
BAB 2: Pemrosesan Citra Digital Sederhana dengan JavaScript

---

1. Tujuan

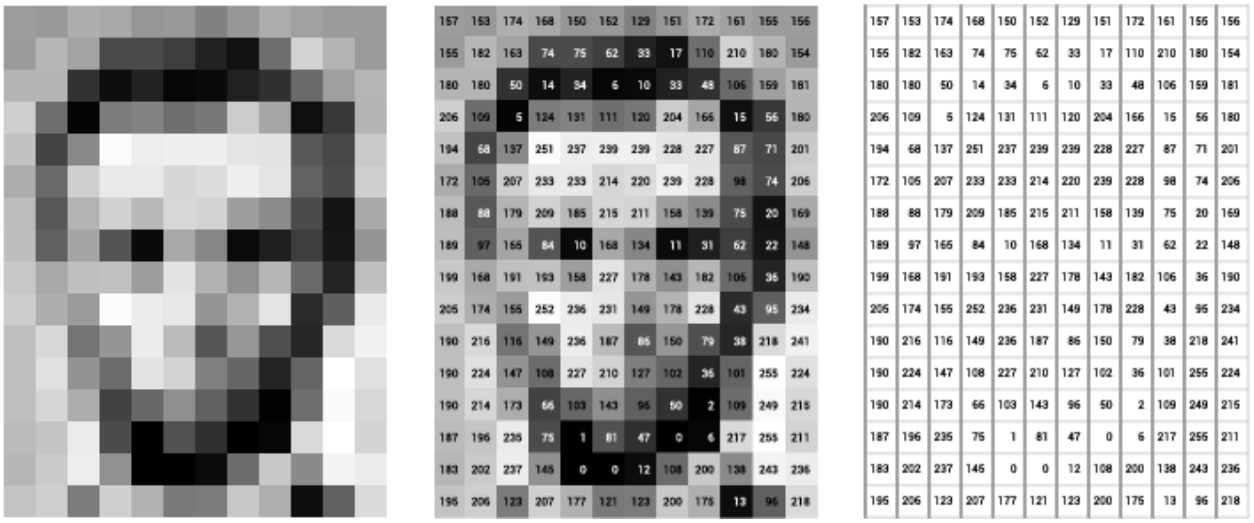
- Memahami dasar-dasar pemrosesan citra digital menggunakan JavaScript dan HTML5 Canvas.
- Menerapkan algoritma pemrosesan citra seperti konversi ke grayscale, deteksi tepi (dengan operator Sobel), efek blur, serta pemisahan kanal warna (RGB).
- Mengembangkan keterampilan pemrograman melalui pembuatan aplikasi interaktif yang memproses gambar secara real time di browser.

---

2. Dasar Teori

2.1. Dasar Citra Digital

Citra digital adalah representasi gambar dalam format digital yang tersusun atas piksel-piksel. Setiap piksel merupakan unit terkecil dengan informasi warna, biasanya direpresentasikan dengan model RGB (Red, Green, Blue) dan terkadang kanal alpha untuk transparansi. Data citra dapat diakses dan dimanipulasi melalui array numerik, di mana setiap empat elemen mewakili komponen R, G, B, dan A pada sebuah piksel. Pada Gambar 1. Ditampilkan nilai piksel pada sebuah gambar digital.



Gambar 1. Piksel pada citra digital (https://ai.stanford.edu/)

2.2. Kernel dan Matriks dalam Pemrosesan Citra

Kernel adalah sebuah matriks kecil yang digunakan untuk melakukan operasi konvolusi pada citra digital. Operasi konvolusi melibatkan pergeseran kernel di atas seluruh piksel gambar, mengalikan nilai piksel dengan nilai kernel, dan menjumlahkan hasilnya.

Contoh:

- **Sobel Kernel:** Digunakan untuk mendeteksi tepi; terdapat dua jenis kernel (Gx untuk deteksi vertikal dan Gy untuk deteksi horizontal).
- **Box Blur Kernel:** Matriks rata-rata yang digunakan untuk menghasilkan efek blur. Konsep kernel sangat penting karena menentukan bagaimana piksel tetangga memengaruhi hasil pengolahan pada tiap piksel, sehingga menghasilkan berbagai efek seperti edge detection, blur, dan sharpen.

-1 0 1 -2 0 2 -1 0 1	0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11
(a) Right Sobel Kernel	(b) Blur Kernel

Gambar 2. Matrik kernel dalam pemrosesan citra

2.3. Konversi ke Grayscale

Konversi citra berwarna ke grayscale dilakukan dengan mengombinasikan nilai RGB menggunakan bobot tertentu (misalnya 0.299 untuk merah, 0.587 untuk hijau, dan 0.114 untuk biru) agar mencerminkan sensitivitas mata manusia terhadap cahaya.
Sumber: citeGrayscaleTheory0

2.4. Deteksi Tepi dengan Operator Sobel

Operator Sobel adalah metode untuk mendeteksi tepi pada citra dengan menerapkan dua kernel:

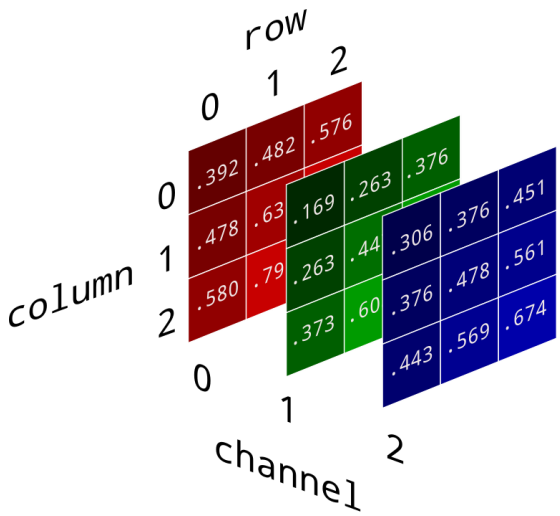
- **Kernel Gx:** Mendeteksi perubahan intensitas secara horizontal (garis vertikal).
- **Kernel Gy:** Mendeteksi perubahan intensitas secara vertikal (garis horizontal).
  Dalam praktik ini, untuk mendeteksi garis kanan dan atas, hasil konvolusi disesuaikan agar hanya menampilkan nilai yang diinginkan (positif untuk kanan, nilai negatif (dikonversi ke absolut) untuk atas).
  Sumber: citeWikipediaSobel0

2.5. Efek Blur dengan Box Filter

Efek blur dicapai dengan menggunakan kernel box (filter rata-rata) yang menghitung rata-rata nilai piksel pada area sekitarnya. Hal ini mengurangi detail tajam dan menghasilkan gambar yang tampak kabur.
Sumber: citeDigitalImageProcessing0

2.6. Pemisahan Kanal Warna (RGB)

Setiap citra berwarna dapat dipisahkan ke dalam tiga komponen: merah, hijau, dan biru. Teknik ini berguna untuk analisis individual masing-masing kanal atau untuk menerapkan efek khusus pada salah satu kanal saja. Pada Gambar ini,



Sumber

Pada modul ini, Anda belajar menerapkan dasar-dasar pemrosesan citra digital dengan hasil akhir seperti gambar berikut

---

3. Alat dan Bahan

- **Komputer** (Windows, macOS, atau Linux).
- **Browser Modern** (misalnya Google Chrome, Mozilla Firefox, atau Microsoft Edge).
- **Text Editor/IDE:**
    - Visual Studio Code
    - Atau alternatif: CodePen.io (untuk eksperimen langsung secara online).
- **File Source Code:**
    - `index.html`
    - `apps.js`
- **Gambar Digital** (JPEG, PNG, dll) sebagai sampel input.

---

4. Langkah Kerja

Langkah 1: Persiapan Proyek

1. Buat sebuah folder proyek baru.
2. Di dalam folder tersebut, buat dua file: `index.html` dan `apps.js`.

Langkah 2: Menulis Kode HTML

Buat file `index.html` dengan konten berikut:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>Aplikasi Pemrosesan Citra Digital Sederhana</title>
  <style>
    body {
      font-family: sans-serif;
      margin: 20px;
    }
    #canvas {
      border: 1px solid #ccc;
      margin-top: 10px;
    }
    .channel-canvas {
      border: 1px solid #ccc;
      margin: 10px;
    }
  </style>
</head>
<body>
  <h1>Aplikasi Pemrosesan Citra Digital</h1>
  <input type="file" id="upload" accept="image/*">
  <br><br>
  <label for="operation">Pilih operasi:</label>
  <select id="operation">
    <option value="grayscale">Ubah warna RGB - Grayscale</option>
    <option value="edgeRight">Deteksi garis kanan</option>
    <option value="edgeTop">Deteksi garis atas</option>
    <option value="blur">Blur</option>
    <option value="separate">Pisahkan Kanal RGB</option>
  </select>
  <button id="process">Proses</button>
  <br><br>
  <canvas id="canvas"></canvas>
  <!-- Kontainer untuk hasil channel terpisah -->
  <div id="channels"></div>

  <script src="apps.js"></script>
</body>
</html>
```

Langkah 3: Menulis Kode JavaScript

Buat dan atau buka file `apps.js` dan bagi penulisan kode menjadi beberapa bagian berikut.

3.1. Kerangka JavaScript

Buat struktur dasar untuk memuat gambar, mengambil data dari canvas, dan memanggil fungsi sesuai pilihan operasi.

Pertama, tambahkan fungsi DOMContentLoaded. Blok fungsi ini akan berfungsi ketika element pada html selesai di load.

```
document.addEventListener('DOMContentLoaded', function() {

  // Load gambar dan tampilkan pada canvas
  upload.addEventListener('change', function(e) {
  });

  // Proses gambar sesuai pilihan operasi
  processButton.addEventListener('click', function() {
  });

  // Fungsi pendukung akan ditambahkan di bawah ini.
  // ...
});
```

Kedua, tambahkan variabel untuk dipasangkan dengan elemen HTML berdasarkan ID elemen.

```
document.addEventListener('DOMContentLoaded', function() {

  const upload = document.getElementById('upload');
  const canvas = document.getElementById('canvas');
  const ctx = canvas.getContext('2d');
  const processButton = document.getElementById('process');
  const operationSelect = document.getElementById('operation');
  const channelsContainer = document.getElementById('channels');
  let originalImageData = null;

 // Load gambar dan tampilkan pada canvas dan seterusnya
});
```

Ketiga, tambahkan fungsi didalam 'DOMContentLoaded' untuk menangani proses unggah gambar

```
 // Load gambar dan tampilkan pada canvas
  upload.addEventListener('change', function(e) {
    const file = e.target.files[0];
    if (!file) return;
    const reader = new FileReader();
    reader.onload = function(event) {
      const img = new Image();
      img.onload = function() {
        canvas.width = img.width;
        canvas.height = img.height;
        ctx.drawImage(img, 0, 0);
        originalImageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
        channelsContainer.innerHTML = ''; // Bersihkan hasil sebelumnya
      }
      img.src = event.target.result;
    }
    reader.readAsDataURL(file);
  });
```

Keempat, tambahkan fungsi berikutnya didalam 'DOMContentLoaded' untuk menangani tombol diklik

```
// Proses gambar sesuai pilihan operasi
  processButton.addEventListener('click', function() {
    if (!originalImageData) return;
    const op = operationSelect.value;
    let resultImageData;
    channelsContainer.innerHTML = ''; // Hapus tampilan channel sebelumnya

    switch(op) {
      case 'grayscale': {
        resultImageData = toGrayscale(new ImageData(new
Uint8ClampedArray(originalImageData.data), originalImageData.width,
originalImageData.height));
        ctx.putImageData(resultImageData, 0, 0);
        break;
      }
      case 'edgeRight': {
        let imgData = toGrayscale(new ImageData(new
Uint8ClampedArray(originalImageData.data), originalImageData.width,
originalImageData.height));
        const kernelRight = [
          -1, 0, 1,
          -2, 0, 2,
          -1, 0, 1
        ];
        resultImageData = convolveGrayscale(imgData, kernelRight, 'right');
        ctx.putImageData(resultImageData, 0, 0);
        break;
      }
      case 'edgeTop': {
        let imgData = toGrayscale(new ImageData(new
Uint8ClampedArray(originalImageData.data), originalImageData.width,
originalImageData.height));
        const kernelTop = [
          -1, -2, -1,
           0,  0,  0,
```

```
          1,  2,  1
        ];
        resultImageData = convolveGrayscale(imgData, kernelTop, 'top');
        ctx.putImageData(resultImageData, 0, 0);
        break;
      }
      case 'blur': {
        const kernelBlur = [
          1/9, 1/9, 1/9,
          1/9, 1/9, 1/9,
          1/9, 1/9, 1/9
        ];
        resultImageData = convolveColor(originalImageData, kernelBlur);
        ctx.putImageData(resultImageData, 0, 0);
        break;
      }
      case 'separate': {
        separateChannels(originalImageData);
        ctx.putImageData(originalImageData, 0, 0);
        break;
      }
      default:
        return;
    }
  });
```

3.2. Fungsi RGB - Grayscale

Fungsi untuk mengkonversi citra berwarna ke grayscale menggunakan rumus bobot:

```
1. function toGrayscale(imageData) {
2.   const data = imageData.data;
3.   for (let i = 0; i < data.length; i += 4) {
4.     const gray = 0.299 * data[i] + 0.587 * data[i+1] + 0.114 * data[i+2];
5.     data[i] = data[i+1] = data[i+2] = gray;
6.   }
7.   return imageData;
8. }
```

3.3. Fungsi Deteksi Garis Kanan

Menggunakan operator Sobel dengan kernel Gx dan menyesuaikan agar hanya nilai positif yang tampil:

```
1.  function convolveGrayscale(imageData, kernel, directional) {
2.    const width = imageData.width;
3.    const height = imageData.height;
4.    const src = imageData.data;
5.    const output = new ImageData(width, height);
6.    const dst = output.data;
7.    const kernelSize = Math.sqrt(kernel.length);
8.    const half = Math.floor(kernelSize / 2);
9.
10.   for (let y = 0; y < height; y++) {
11.     for (let x = 0; x < width; x++) {
12.       let sum = 0;
13.       for (let ky = -half; ky <= half; ky++) {
14.         for (let kx = -half; kx <= half; kx++) {
15.           const posX = x + kx;
16.           const posY = y + ky;
17.           if (posX >= 0 && posX < width && posY >= 0 && posY < height) {
18.             const i = (posY * width + posX) * 4;
19.             const pixel = src[i]; // Karena sudah grayscale, R=G=B
20.             const k = kernel[(ky + half) * kernelSize + (kx + half)];
21.             sum += pixel * k;
22.           }
23.         }
24.       }
25.       const idx = (y * width + x) * 4;
26.       if (directional === 'right') {
27.         sum = sum > 0 ? sum : 0;
28.       } else if (directional === 'top') {
29.         sum = sum < 0 ? -sum : 0;
30.       } else {
31.         sum = Math.abs(sum);
32.       }
33.       dst[idx] = dst[idx+1] = dst[idx+2] = (sum > 255 ? 255 : sum);
34.       dst[idx+3] = 255;
35.     }
36.   }
37.   return output;
38. }
```

3.4. Fungsi Deteksi Garis Atas

Fungsi ini menggunakan kode yang sama dengan deteksi garis kanan, namun dengan parameter `directional` bernilai `'top'`.

Catatan: Lihat kode pada fungsi `convolveGrayscale` di atas.

3.5. Fungsi Blur

Menerapkan konvolusi pada citra berwarna menggunakan kernel box blur:

```
function convolveColor(imageData, kernel) {
  const width = imageData.width;
  const height = imageData.height;
  const src = imageData.data;
  const output = new ImageData(width, height);
  const dst = output.data;
  const kernelSize = Math.sqrt(kernel.length);
  const half = Math.floor(kernelSize / 2);

  for (let y = 0; y < height; y++) {
    for (let x = 0; x < width; x++) {
      let sumR = 0, sumG = 0, sumB = 0;
      for (let ky = -half; ky <= half; ky++) {
        for (let kx = -half; kx <= half; kx++) {
          const posX = x + kx;
          const posY = y + ky;
          if (posX >= 0 && posX < width && posY >= 0 && posY < height) {
            const i = (posY * width + posX) * 4;
            const k = kernel[(ky + half) * kernelSize + (kx + half)];
            sumR += src[i] * k;
            sumG += src[i+1] * k;
            sumB += src[i+2] * k;
          }
        }
      }
      const idx = (y * width + x) * 4;
      dst[idx]   = Math.min(Math.max(sumR, 0), 255);
      dst[idx+1] = Math.min(Math.max(sumG, 0), 255);
      dst[idx+2] = Math.min(Math.max(sumB, 0), 255);
      dst[idx+3] = 255;
    }
  }
  return output;
}
```

3.6. Fungsi Pemisahan Kanal RGB

Fungsi ini membuat tiga objek `ImageData` untuk masing-masing kanal (merah, hijau, dan biru) dan menampilkannya pada canvas terpisah:

```
1. function separateChannels(imageData) {
2.   const width = imageData.width;
3.   const height = imageData.height;
4.   const src = imageData.data;
5.
6.   const redData = new ImageData(width, height);
7.   const greenData = new ImageData(width, height);
8.   const blueData = new ImageData(width, height);
9.
10.  for (let i = 0; i < src.length; i += 4) {
11.    // Red channel: tampilkan komponen merah saja
12.    redData.data[i] = src[i];
13.    redData.data[i+1] = 0;
14.    redData.data[i+2] = 0;
15.    redData.data[i+3] = src[i+3];
16.
17.    // Green channel: tampilkan komponen hijau saja
18.    greenData.data[i] = 0;
19.    greenData.data[i+1] = src[i+1];
20.    greenData.data[i+2] = 0;
21.    greenData.data[i+3] = src[i+3];
22.
23.    // Blue channel: tampilkan komponen biru saja
24.    blueData.data[i] = 0;
25.    blueData.data[i+1] = 0;
26.    blueData.data[i+2] = src[i+2];
27.    blueData.data[i+3] = src[i+3];
28.  }
29.
30.  // Fungsi untuk membuat canvas baru dan menampilkan ImageData per channel
31.  function createChannelCanvas(channelName, data) {
32.    const channelCanvas = document.createElement('canvas');
33.    channelCanvas.width = width;
34.    channelCanvas.height = height;
35.    channelCanvas.className = 'channel-canvas';
36.    const context = channelCanvas.getContext('2d');
37.    context.putImageData(data, 0, 0);
38.    // Tambahkan judul untuk masing-masing channel
39.    const title = document.createElement('p');
40.    title.innerText = channelName;
41.    const container = document.createElement('div');
42.    container.style.display = 'inline-block';
43.    container.style.textAlign = 'center';
44.    container.style.marginRight = '10px';
45.    container.appendChild(title);
46.    container.appendChild(channelCanvas);
47.    return container;
48.  }
49.
50.  // Tampilkan canvas untuk masing-masing channel di dalam container
51.  channelsContainer.appendChild(createChannelCanvas('Red Channel', redData));
52.  channelsContainer.appendChild(createChannelCanvas('Green Channel', greenData));
53.  channelsContainer.appendChild(createChannelCanvas('Blue Channel', blueData));
54.}
```

Catatan: Pastikan seluruh fungsi pendukung (poin 3.2 sampai 3.6) ditempatkan di dalam kerangka event listener `DOMContentLoaded` atau diimpor secara terpisah sehingga dapat diakses oleh kode utama.

Langkah 4: Pengujian Aplikasi

1. Buka file `index.html` di browser menggunakan Visual Studio Code (dengan Live Server) atau unggah ke CodePen.io.
2. Pilih sebuah gambar melalui input file.
3. Pilih salah satu operasi dari dropdown (misalnya: grayscale, deteksi garis kanan, deteksi garis atas, blur, atau pisahkan kanal RGB).
4. Klik tombol **Proses** untuk melihat hasilnya pada canvas (dan tampilan terpisah untuk kanal RGB jika opsi tersebut dipilih).

Langkah 5: Eksperimen dan Analisis

1. Uji setiap filter dan perhatikan perbedaan hasil dari masing-masing operasi.
2. Cobalah mengubah nilai-nilai kernel (misalnya pada efek blur atau deteksi tepi) untuk mengamati perubahan hasil citra.

---

5. Tugas

1. **Pengembangan Fitur:**
    - Tambahkan filter baru, seperti efek sharpen atau deteksi tepi dengan operator Prewitt.
    - Kembangkan fitur untuk menggabungkan dua atau lebih efek secara berurutan (misalnya, blur kemudian deteksi tepi).
2. **Analisis Teoritis:**
    - Jelaskan perbedaan mendasar antara algoritma deteksi tepi dengan kernel Sobel dan Prewitt.
    - Diskusikan bagaimana perubahan nilai-nilai dalam kernel memengaruhi hasil pengolahan citra.
3. **Peningkatan Antarmuka:**
    - Rancang ulang tampilan antarmuka agar lebih interaktif dan informatif dengan CSS dan JavaScript.
    - Tambahkan fitur untuk mengunduh hasil citra yang telah diproses.
4. **Refleksi Praktikum:**
    - Buat laporan singkat mengenai kesulitan yang ditemui selama praktikum beserta solusi yang dicoba.
    - Sertakan analisis perbandingan hasil tiap filter serta pemisahan kanal RGB.

Sumber-sumber:

- citeMDNCanvas0 – MDN Web Docs: Canvas API
- citeWikipediaSobel0 – Wikipedia: Sobel operator
- citeGrayscaleTheory0 – Literatur dasar pemrosesan citra (misalnya buku "Digital Image Processing")
- citeDigitalImageProcessing0 – Literatur dan referensi teori blur dan konvolusi dalam pemrosesan citra